# Code Revising

**Numpy:**

Import numpy as np

a = np.array([0, 1, 2, 3, 4])

a

print(np.\_\_version\_\_)

a.dtype

type(a)

arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[1:5:2])

a.size

a.ndim

a.shape

Numpy Statistcal functions

a = np.array([1, -1, 1, -1])

mean = a.mean()

mean

standard\_deviation=a.std()

standard\_deviation

max\_b = b.max()

max\_b

min\_b = b.min()

min\_b

Operations

z = np.add(u, v)

z

arr3 = np.subtract(arr1, arr2)

z = np.multiply(x, y)

z

c = np.divide(a, b)

c

arr3 = np.dot(arr1, arr2)

Mathematical Functions

np.pi

np.sin(x)

np.linspace(-2, 2, num=5)

result = array([-2., -1., 0., 1., 2.])

2D numpy Array

a = [[11, 12, 13], [21, 22, 23], [31, 32, 33]]

a

A[0][0]

array([11, 12]) #Result

A[0][0:2]

array([11, 12]) #Result

A[0:2, 2]

array([13, 23]) #Result

# Get the transposed of C

C.T

array([[1, 2, 3],

[1, 2, 3]]) #Result

**Pandas:**

import pandas as pd

x = {'Name': ['Rose','John', 'Jane', 'Mary'], 'ID': [1, 2, 3, 4], 'Department': ['Architect Group', 'Software Group', 'Design Team', 'Infrastructure'],

      'Salary':[100000, 80000, 50000, 60000]}

#print(x["Name"])

#casting the dictionary to a DataFrame

df = pd.DataFrame(x)

#display the result df

df

To select column

x = df[['ID']]

x

z = df[['Department','Salary','ID']]

z

loc() : is a label-based data selecting method which means that we have to pass the name of the row or column that we want to select. This method includes the last element of the range passed in it.

Simple syntax for your understanding:

 - loc[row\_label, column\_label]

Iloc(): is an indexed-based selecting method which means that we have to pass an integer index in the method to select a specific row/column. This method does not include the last element of the range passed in it.

Simple syntax for your understanding:

 - iloc[row\_index, column\_index]

# Access the value on the first row and the first column

df.iloc[0, 0]

Read carefully the comment. First row and first column, not first element and first nested element. Accessing dataframe is different compared to list, tuple…

# Access the column using the name

df.loc[0, 'Salary']

We can use an integer to specified the index of the row with loc[] function as well.

df2=df

df2=df2.set\_index("Name")

#To display the first 5 rows of new dataframe

df2.head()

#Now, let us access the column using the name

df2.loc['Jane', 'Salary']

print(df)

df.iloc[0:2, 0:3]

print(df)

df.loc[0:2,'ID':'Department']

loc() start and end indexes are inclusive while iLoc() end index is exclusive.

**Loading Data with Pandas**

Check the note

**Reading Files**

# Read the Example1.txt

example1 = "example1.txt"

file1 = open(example1, "r")

file1.name

file1.mode

FileContent = file1.read()

FileContent

file1.close()

The common and better way to open and read a file

# Open file using with

with open(example1, "r") as file1:

    FileContent = file1.read()

    print(FileContent)

# Read first four characters

with open(example1, "r") as file1:

    print(file1.read(4))

with open(example1, "r") as file1:

    print(file1.read(4))

    print(file1.read(4))

    print(file1.read(7))

    print(file1.read(15))

# Read one line

with open(example1, "r") as file1:

    print("first line: " + file1.readline())

with open(example1, "r") as file1:

    print(file1.readline(20)) # does not read past the end of line

    print(file1.read(20)) # Returns the next 20 chars

with open(example1,"r") as file1:

        i = 0

        for line in file1:

            print("Iteration", str(i), ": ", line)

            i = i + 1

with open(example1, "r") as file1:

    FileasList = file1.readlines()

**Write Files**

exmp2 = 'example2.txt'

with open(exmp2, 'w') as writefile:

    writefile.write("This is line A")

with open(exmp2, 'w') as writefile:

    writefile.write("This is line A\n")

    writefile.write("This is line B\n")

with open('Example2.txt', 'w') as writefile:

    writefile.write("Overwrite\n")

with open('Example2.txt', 'r') as testwritefile:

    print(testwritefile.read())

Appending Files

with open('Example2.txt', 'a') as testwritefile:

    testwritefile.write("This is line C\n")

    testwritefile.write("This is line D\n")

    testwritefile.write("This is line E\n")

with open('Example2.txt', 'a+') as testwritefile:

    testwritefile.write("This is line E\n")

    print(testwritefile.read())

with open('Example2.txt', 'a+') as testwritefile:

    print("Initial Location: {}".format(testwritefile.tell()))

    data = testwritefile.read()

    if (not data):  #empty strings return false in python

            print('Read nothing')

    else:

            print(testwritefile.read())

    testwritefile.seek(0,0) # move 0 bytes from beginning.

    print("\nNew Location : {}".format(testwritefile.tell()))

    data = testwritefile.read()

    if (not data):

            print('Read nothing')

    else:

            print(data)

    print("Location after read: {}".format(testwritefile.tell()) )

with open('Example2.txt', 'r+') as testwritefile:

    testwritefile.seek(0,0) #write at beginning of file

    testwritefile.write("Line 1" + "\n")

    testwritefile.write("Line 2" + "\n")

    testwritefile.write("Line 3" + "\n")

    testwritefile.write("Line 4" + "\n")

    testwritefile.write("finished\n")

    testwritefile.seek(0,0)

    print(testwritefile.read())

with open('Example2.txt', 'r+') as testwritefile:

    testwritefile.seek(0,0) #write at beginning of file

    testwritefile.write("Line 1" + "\n")

    testwritefile.write("Line 2" + "\n")

    testwritefile.write("Line 3" + "\n")

    testwritefile.write("Line 4" + "\n")

    testwritefile.write("finished\n")

    #Uncomment the line below

    testwritefile.truncate()

    testwritefile.seek(0,0)

    print(testwritefile.read())

# Copy file to another

with open('Example2.txt','r') as readfile:

    with open('Example3.txt','w') as writefile:

          for line in readfile:

                writefile.write(line)